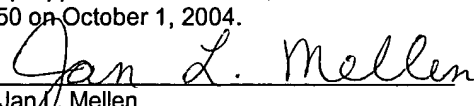




AF/2126
JFW

TRANSMITTAL OF APPEAL BRIEF		Docket No. ST9-98-083
Applicant:	Stephen Lewallen	
Serial No:	09/244,291	
Filed:	February 3, 1999	
For:	METHOD AND APPARATUS FOR PROVIDING PROTOCOL INDEPENDENT NAMING AND LIFE CYCLE SERVICES IN AN OBJECT- ORIENTED SYSTEM	
Examiner:	The T. Ho	
Art Unit:	2126	

<p align="center">CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)</p> <p>The undersigned hereby certifies that this document is being placed in the United States mail with first-class postage attached, addressed to Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450 on October 1, 2004.</p> <p align="right"> Jan L. Mellen</p>

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Transmitted herewith is the APPEAL BRIEF in this application, with respect to the Notice of Appeal filed on August 4, 2004.

Status of Applicant

This application is on behalf of International Business Machines Corporation.

☐ Applicant claims small entity status.

Extension of Time

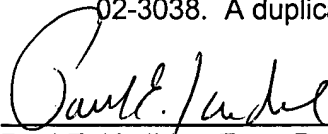
☐ The proceedings herein are for a patent application and the provisions of 37 C.F.R.1.136 apply. An extension of time of months is requested.

Fee Due

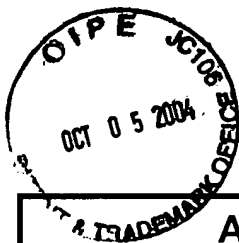
Appeal Brief Fee 340.00
Extension of Time Fee
TOTAL FEE \$340.00

Payment

- ☒ Enclosed is a check in the amount of the total fee.
☐ The Commissioner is authorized to charge the total fee to Deposit Account No. 02-3038.
☒ The Commissioner is hereby authorized to charge any other fees under 37 C.F.R. §1.16 and §1.17 that may be required, or credit any overpayment, to Deposit Account No. 02-3038. A duplicate of this transmittal letter is attached.

Date: 10/1/04

Paul E. Kudirka, Esq. Reg. No. 26,931
KUDIRKA & JOBSE, LLP
Customer Number 021127
Tel: (617) 367-4600 Fax: (617) 367-4656



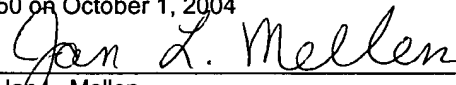
**APPEAL BRIEF UNDER
37 CFR §41.37**

Docket No. ST9-98-083

Applicant: Stephen Lewallen
Serial No: 09/244,291
Filed: February 3, 1999
For: METHOD AND APPARATUS FOR PROVIDING PROTOCOL
INDEPENDENT NAMING AND LIFE CYCLE SERVICES IN AN OBJECT-
ORIENTED SYSTEM
Examiner: The T. Ho
Art Unit: 2126

CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)

The undersigned hereby certifies that this document is being placed in the United States mail with first-class postage attached, addressed to Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450 on October 1, 2004


Jan L. Mellen

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

This brief is in furtherance of the Notice of Appeal, filed in this case on August 4, 2004.

The fees required under § 41.20(b)(2), and any required petition for extension of time for filing this brief and fees therefor, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

A single copy of this brief is transmitted. (37 C.F.R. 41.37(a)) and contains these items under the following headings, and in the order set forth below (37 C.F.R. 41.37(c)(1)):

- I REAL PARTY IN INTEREST
- II RELATED APPEALS AND INTERFERENCES
- III STATUS OF CLAIMS
- IV STATUS OF AMENDMENTS
- V SUMMARY OF CLAIMED SUBJECT MATTER
- VI GROUNDS OF REJECTION
- VII ARGUMENT
- VIII CLAIMS APPENDIX

I REAL PARTY IN INTEREST (37 C.F.R. 41.37(c)(1)(i))

The real party in interest in this appeal is International Business Machines Corporation.

II RELATED APPEALS AND INTERFERENCES (37 C.F.R. 41.37(c)(1)(ii))

There are no other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal.

III STATUS OF CLAIMS (37 C.F.R. 41.37(c)(1)(iii))

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-36

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims pending: 1-36
2. Claims canceled: none
3. Claims withdrawn from consideration, but not canceled: none
4. Claims allowed: none
5. Claims rejected: 1-36

C. CLAIMS ON APPEAL

The claims on appeal are: 1-36

IV STATUS OF AMENDMENTS (37 C.F.R. 41.37(c)(1)(iv))

All amendments submitted by applicant have been entered. The Request for Reconsideration filed on February 17, 2004 was considered by the examiner as indicated in the office response dated May 4, 2004.

V SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R.41.37(c)(1)(v))

The present invention relates to an object-oriented system that uses distributed storage to store various objects on a set of computers connected by a network. In such a system, an object can be stored in a persistent repository 1320 by a server program on one computer, retrieved and manipulated by an application program 1300 in another computer and then stored again on the first computer. The invention is an object that is part of an application which

provides life cycle services, such as the creation, modification and destruction of objects. The inventive object is called a "moniker" object constructed from a moniker class 625 (Figure 6) and it assists an application program in storing and retrieving distributed objects so that the life cycle services system 1310, (Figure 13), located in the memory 1308, can interact with the distributed objects and provide life cycle services (page 5, lines 17-25 and page 15, line 1 – page 16, line 10).

Normally, when an application program stores an object in local storage 1328, that object, including the object data and methods is streamed into the local storage 1328 where it can be maintained by the local life cycle services system 1310. However, in a distributed system, the application program may be using an object that is actually stored in a remote repository 1320, although the application 1300 may not be aware of this. From the application's point of view, it is performing a local storage operation and expects the local life cycle services system 1310 to manage the object. In order to allow the local life cycle services system 1310 to appear to manage the remote object, in accordance with the invention, a first stream object (TalonOutputStream Object, see page 10, line 27 – page 11, line 8) is created from an output stream class 450 (Figure 4) to during the streaming of the object into local storage, a moniker object is automatically substituted for the actual distributed object (see Figures 13 and 14 and accompanying description at page 22, line 3 – page 23, line 3.) Thus, the moniker object is stored in local storage in place of the real object. The moniker object in the local storage can then be maintained by the inventive life cycle services system. The actual distributed object can be stored in the remote repository 1320. This happens transparently to the application program 1300 which sees the object as being persisted in local storage 1328.

Similarly, when an application program attempts to retrieve an object from local storage 1528, the moniker object in local storage 1528 is instead streamed in from local storage 1528 by a second stream object (TalonInputStream object instantiated from an InputStream class 440, page 11, lines 3-8). During this streaming, a reference to the real distributed object is substituted for the moniker object in the local memory 1508. The real object is then created in the remote computer. (See Figures 15 and 16 and accompanying description at page 23, lines 4-29). This transparent substitution allows the lifecycle services program 1310, which is located in the local computer to interact, with a local moniker object to provide the lifecycle services without modifying the application program to take into account the fact that the distributed object is not locally stored.

The moniker object of the present invention can also be used to locate the distributed object. To this end, the inventive system maintains a runtime repository database 1116 of moniker name-object reference pairs which it can use to look up a reference to the distributed object. If a moniker name is presented to this database and an object reference is not found, then the inventive system creates a directory service adapter object 1112 and uses it to query an existing directory service 1114 in an attempt to locate the distributed object. (See Figures 11 and 12 and the accompanying description at page 20, line 27 – page 22, line 2).

VI GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL (37 C.F.R. 41.37(c)(1)(vi))

Claims 1-36 stand rejected under 35 U.S.C. §103(a) as obvious over U.S. Patent No. 6,263,379 (Atkinson) in view of U.S. Patent No. 6,460,058 (Koppolu).

VII ARGUMENT (37 C.F.R. 41.37(c)(1)(vii))

Prima facie obviousness has not been established because the combination of Atkinson and Koppolu does not teach or suggest the structure recited in claims 1, 2, 5, 7, 9-12, 15, 17, 19-22, 25, 27 and 29-30.

Obviousness is a legal conclusion based on factual evidence. Graham v. John Deere Co. 383 US 1, 148 USPQ 459 (1966). The PTO has the burden under section 103 to establish a prima facie case of obviousness. In re Piasecki, 745 F.2d 1468, 1471-72, 223 USPQ 785, 787-87 (Fed. Cir. 1984). To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1970.)

Both the Atkinson and the Koppolu references disclose "moniker" objects. However, although these objects have the same name as the moniker objects of the present invention, they have a different construction and function in an entirely different manner. Neither Atkinson nor Koppolu mention life cycle services and neither reference is concerned with the problem solved by the present invention. The moniker objects disclosed in both Atkinson and Koppolu are designed to assist a system that must work with diverse objects, each of which has its own interface. Atkinson discusses this problem in connection with application programs, such as word processing systems, that must operate on compound documents. These compound

documents may contain linked or embedded data objects, such as spreadsheet objects, that have diverse interfaces. In the Atkinson system, a moniker object is used to provide a link to the data objects. Different moniker objects can be used to link to different types of objects, such as files, items and pointers. The advantage is that the moniker object has a consistent interface IMoniker. Thus, the application program can interact in a consistent way with the moniker object and the moniker object controls the real objects.

Koppolu uses its moniker objects in a similar fashion, but in the context of a browser in order to manage hyperlinks to non-HTML documents. Again, the moniker object has a consistent interface IMoniker with which the browser operates. Thus, the browser can interact in a consistent way with the moniker object and the moniker object controls the real objects that are hyperlinked.

In both the Atkinson and Koppolu references, the moniker object itself can be stored (persisted) and retrieved (resurrected.) In addition, in both references, the disclosed moniker objects (and the interface IMoniker) contain functions (BindToObject and BindToStorage) that allow the moniker object to store and retrieve the object to which the moniker object is bound. However, during the storage of the bound object, the moniker object is not substituted for the bound object so that the moniker object is stored in place of the bound object. For example, the moniker object can be used to store its bound object using the BindToStorage function. Then, the moniker object can itself be stored. This, results in both the moniker object and the bound object residing in local storage. If the moniker object was stored in place of the bound object, then only the moniker object would remain in storage after the store operation was complete.

The combination of Atkinson and Koppolu does not teach or suggest the limitations recited in claims 1, 2, 5, 7, 9-12, 15, 17, 19-22, 25, 27 and 29-30. Claim 1 is representative of these claims and explicitly recites that the moniker object is substituted for the distributed object in the local storage during the storage operations so that the moniker object is stored in place of the distributed object. For example, claim 1, in lines 7-10, recites "a first stream object which automatically substitutes the moniker object for the distributed object during the streaming of the distributed object out from the memory to the local storage so that the moniker object is stored in the local storage in place of the distributed object." As set forth above, this operation does not occur in either of the cited references. Since neither reference shows this operation, the

combination of the references cannot teach or suggest the operation. Consequently, claim 1 patentably distinguishes over the cited references.

The examiner acknowledges that Atkinson does not teach substituting the moniker object for the distributed object as recited in claim 1. However, the examiner claims that the Koppolu reference and, specifically, the BindToObject function of the IMoniker interface, teaches substituting the moniker object for the distributed object. The BindToObject function is described in Koppolu at column 14, line 64 to column 15, line 4. There, Koppolu states that the BindToObject function performs binding by instantiating the named object in the memory of the computer and returning a pointer to the interface of the named object to the client. Thus, the BindToObject function is involved in retrieving an object from storage into memory rather than storing an object from memory into storage as recited in claim 1. Accordingly, it will be discussed below with regard to the Group 2 claims that deal with retrieving an object from storage. Instead, the BindToStorage function is called by a client to store an object in memory to local storage. At column 15, lines 4-8, the BindToStorage function is described. Koppolu states that the BindToStorage function performs binding by instantiating the named object onto an OLE storage stream which is stored in the computer's secondary storage rather than into main memory. It is quite clear that this operation is considerably different from that claimed in claim 1. First, the result of the BindToStorage function is that the named object, not the moniker object is stored in the local storage. Thus, the moniker object is not "stored in the local storage in place of the distributed object" as recited in claim 1. Further, the storage operation is caused when a client calls the BindToStorage function and is not performed by automatically substituting the moniker object for the distributed object during the streaming of the distributed object out from the memory to the local storage as recited in claim 1. Thus, the combination of Atkinson and Koppolu explicitly pointed out by the examiner does not teach or suggest the limitations recited in claim 1.

Prima facie obviousness has not been established because the combination of Atkinson and Koppolu does not teach or suggest the structure recited in claims 3-4, 13-14 and 23-24.

Claims 3-4, 13-14 and 23-24 are concerned with retrieving an object from local storage. Claim 3 is representative and recites "a second stream object which automatically substitutes a reference to the distributed object for the moniker object during the streaming of the moniker

object in from the local storage to the memory so that a reference to the distributed object is created in memory in place of the moniker object." Thus, the second stream object retrieves a distributed object from "local" storage into memory by retrieving the moniker object from local storage, but then substituting a reference to the distributed object for the moniker object during the streaming of the moniker object into memory. This operation results in only the distributed object residing in memory.

The examiner points to the IPersistStream disclosed in Atkinson at column 14, lines 37-65 as teaching a reference to the distributed object for the moniker object. First, it is noted that the IPersistStream is used in storing and retrieving the Atkinson moniker object itself. See Atkinson, column 14, lines 37-42. In order to use the Atkinson moniker object in a similar fashion to the claimed moniker object, the client would have to store the moniker object in local storage and then retrieve the moniker object and call the BindToObject method to instantiate the named object. This would certainly not be transparent as is the present invention. First, the client would have to be rewritten so that it stores and retrieves the moniker object instead of the named object. Then, after retrieving the moniker object, the client would have to explicitly call the BindToObject function. This results in both the moniker object and the bound object residing in memory. The distributed object is not "created in memory in place of the moniker object" as recited in claim 3. Further, the Atkinson process would not automatically substitute a reference to the distributed object for the moniker object during the streaming of the moniker object in from the local storage to the memory as recited in claim 3. Since the Koppolu reference discloses the same IPersistStream object as Atkinson, its combination with Atkinson does not alter or extend Atkinson's teachings. Thus, claim 3 patentably distinguishes over the cited combination of references.

Prima facie obviousness has not been established because the combination of Atkinson and Koppolu does not teach or suggest the structure recited in claims 6, 16 and 26.

Claims 6, 16 and 26 relate to a life cycle services object that manages the life cycle of the distributed object. Claim 6 is representative. The examiner claims that the CreateGenericComposite function disclosed in Atkinson, column 22, lines 37 et seq. discloses such a life cycle services object that can manage a distributed object in accordance with a life cycle services policy. However, the CreateGenericComposite function creates a composite

moniker object from other moniker objects. While it might disclose creating a moniker object in accordance with a predefined policy, claim 6 recites that the life cycle services object manages the **distributed** object, not the moniker object. Koppolu discloses a similar CreateGenericComposite function. Therefore, its combination with Atkinson does not alter or extend Atkinson's teachings. Thus, claim 6 patentably distinguishes over the cited combination of references.

Prima facie obviousness has not been established because the combination of Atkinson and Koppolu does not teach or suggest the structure recited in claims 8, 18 and 28.

Claims 8, 18 and 28 relate to location of a distributed object using an existing directory service. Claim 8 is representative. The examiner claims that Atkinson, at column 14, lines 34-47, teaches a directory service factory that responds to the moniker name by applying the moniker name to an existing directory service. However, at the recited column and lines, Atkinson actually states that each reference to a moniker object that is stored in, for example, a document contains a class ID and a reference to a named object. When a word processing program wants to display the document, it uses the class ID in the reference to access a moniker object factory to create the moniker object and then calls functions in the moniker object along with the reference to the named object to manipulate the named object. Directory services are not mentioned or needed since each reference to a moniker object will have a class ID for that object. Koppolu discloses a similar operation. Therefore, its combination with Atkinson does not alter or extend Atkinson's teachings. Thus, claim 8 patentably distinguishes over the cited combination of references.

Prima facie obviousness has not been established because the combination of Atkinson and Koppolu does not teach or suggest the structure recited in claims 31-36.

Claims 31-36 recite that the persistent repository in which the distributed object is stored is different from the local storage in which the moniker object is stored. Claim 31 is representative. The examiner points to Koppolu Fig. 1 as indicating a persistent repository (Main Memory 40) that is different from local storage (Secondary Storage 42). However, it is clear that Main Memory 40 corresponds to the "memory" recited in the claims and that the "local storage" recited in the claims corresponds to Koppolu's Secondary Storage 42. It is noted that the Main Memory 40 is neither persistent nor located remotely from the local storage as recited

in the claims. Atkinson does not discuss a persistent repository or where this might be located. Therefore, its combination with Koppolu does not alter or extend Koppolu's teachings. Thus, claim 31 patentably distinguishes over the cited combination of references.

Respectfully submitted



Date: 10/1/04

Paul E. Kudirka, Esq. Reg. No. 26,931
KUDIRKA & JOBSE, LLP
Customer Number 021127
Tel: (617) 367-4600 Fax: (617) 367-4656

VIII APPENDIX OF CLAIMS (37 C.F.R. 41.37(c)(1)(viii))

The text of the claims involved in the appeal is:

- 1 1. Apparatus for use with a computer system having a memory, a local storage and an
2 existing directory service operating in the memory, the apparatus providing naming and
3 life cycle services for a distributed object and comprising:
4 a moniker object which contains an identifier that universally identifies an
5 instance of the distributed object and a moniker name; and
6 a first stream object which automatically substitutes the moniker object for the
7 distributed object during the streaming of the distributed object out from the memory to
8 the local storage so that the moniker object is stored in the local storage in place of the
9 distributed object.
- 1 2. Apparatus according to claim 1 wherein the first stream object substitutes the moniker
2 object for the distributed object when the distributed object is persisted.
- 1 3. Apparatus according to claim 1 further comprising a second stream object which
2 automatically substitutes a reference to the distributed object for the moniker object
3 during the streaming of the moniker object in from the local storage to the memory so
4 that a reference to the distributed object is created in memory in place of the moniker
5 object.
- 1 4. Apparatus according to claim 3 wherein the second stream object substitutes the
2 moniker object for the distributed object when the distributed object is resurrected.

- 1 5. Apparatus according to claim 1 wherein life cycle services are provided by associating
2 with the moniker object a predefined policy which specifies how and when life cycle
3 services are performed.
- 1 6. Apparatus according to claim 5 further comprising a life cycle services object which
2 responds to the predefined policy by controlling the life cycle of the distributed object.
- 1 7. Apparatus according to claim 1 further comprising a runtime repository which includes a
2 database of moniker name-object reference pairs.
- 1 8. Apparatus according to claim 7 further comprising a directory service factory object
2 which responds to the moniker name by instantiating a directory service adapter object
3 for applying the moniker name to the existing directory service when the runtime
4 repository does not contain the moniker name.
- 1 9. Apparatus according to claim 1 wherein the distributed object is instantiated in
2 accordance with an object model and the apparatus comprises an object model adapter
3 which processes distributed objects instantiated with the object model.
- 1 10. Apparatus according to claim 9 wherein the object model adapter returns a reference to
2 the distributed object together with a moniker object associated with the distributed
3 object.

1 11. A method for use with a computer system having a memory, a local storage and an
2 existing directory service operating in the memory, the method providing naming and life
3 cycle services for a distributed object and comprising the steps of:
4 (a) instantiating a moniker object which contains an identifier that universally
5 identifies an instance of the distributed object and a moniker name; and
6 (b) using a first stream object to automatically substitute the moniker object for the
7 distributed object during the streaming of the distributed object out from the
8 memory to the local storage so that the moniker object is stored in the local
9 storage in place of the distributed object.

1 12. A method according to claim 11 wherein step (b) comprises the step of:
2 (b1) using the first stream object to substitute the moniker object for the distributed
3 object when the distributed object is persisted.

1 13. A method according to claim 11 further comprising the step of:
2 (c) using a second stream object to automatically substitute a reference to the
3 distributed object for the moniker object during the streaming of the moniker
4 object in from the local storage to the memory so that a reference to the
5 distributed object is created in memory in place of the moniker object.

1 14. A method according to claim 13 wherein step (c) comprises the step of:
2 (c1) using the second stream object to substitute the moniker object for the distributed
3 object when the distributed object is resurrected.

1 15. A method according to claim 11 further comprising the step of:
2 (d) associating with the moniker object a predefined policy which specifies how and
3 when life cycle services are performed.

1 16. A method according to claim 15 further comprising the step of:
2 (e) instantiating a life cycle services object which responds to the predefined policy
3 by controlling the life cycle of the distributed object.

1 17. A method according to claim 11 further comprising the step of:
2 (f) creating a runtime repository which includes a database of moniker name-object
3 reference pairs.

1 18. A method according to claim 17 further comprising the step of:
2 (g) instantiating a directory service factory object which responds to the moniker
3 name by instantiating a directory service adapter object for applying the moniker
4 name to the existing directory service when the runtime repository does not
5 contain the moniker name.

1 19. A method according to claim 11 wherein the distributed object is instantiated in
2 accordance with an object model and wherein the method comprises the step of:
3 (h) instantiating an object model adapter which processes distributed objects
4 instantiated with the object model.

- 1 20. A method according to claim 19 wherein step (h) comprises the step of:
- 2 (h1) returning a reference to the distributed object together with a moniker object
- 3 associated with the distributed object.
- 1 21. A computer program product for use with a computer system having a memory, a local
- 2 storage and an existing directory service operating in the memory, the computer
- 3 program product providing naming and life cycle services for a distributed object and
- 4 comprising a computer usable medium having computer readable program code thereon
- 5 including:
- 6 class code for instantiating a moniker object which contains an identifier that
- 7 universally identifies an instance of the distributed object and a moniker name; and
- 8 class code for instantiating a first stream object which automatically substitutes
- 9 the moniker object for the distributed object during the streaming of the distributed object
- 10 out from the memory to the local storage so that the moniker object is stored in the local
- 11 storage in place of the distributed object.
- 1 22. A computer program product according to claim 21 wherein the class code for
- 2 instantiating a first stream object comprises method code for substituting the moniker
- 3 object for the distributed object when the distributed object is persisted.
- 1 23. A computer program product according to claim 21 further comprising class code for
- 2 instantiating a second stream object which automatically substitutes a reference to the
- 3 distributed object for the moniker object during the streaming of the moniker object in

4 from the local storage to the memory so that a reference to the distributed object is
5 created in memory in place of the moniker object.

1 24. A computer program product according to claim 23 wherein the class code for
2 instantiating the second stream object includes method code for substituting the moniker
3 object for the distributed object when the distributed object is resurrected.

1 25. A computer program product according to claim 21 wherein the class code for
2 instantiating the moniker object further comprises a method for associating with the
3 moniker object a predefined policy which specifies how and when life cycle services are
4 performed.

1 26. A computer program product according to claim 25 further comprising class code for
2 instantiating a life cycle services object which responds to the predefined policy by
3 controlling the life cycle of the distributed object.

1 27. A computer program product according to claim 21 further comprising program code for
2 creating a runtime repository which includes a database of moniker name-object
3 reference pairs.

1 28. A computer program product according to claim 27 further comprising class code for
2 instantiating a directory service factory object which responds to the moniker name by
3 instantiating a directory service adapter object for applying the moniker name to the

4 existing directory service when the runtime repository does not contain the moniker
5 name.

1 29 A computer program product according to claim 21 wherein the distributed object is
2 instantiated in accordance with an object model and wherein the computer program
3 product comprises class code for instantiating an object model adapter which processes
4 distributed objects instantiated with the object model.

1 30. A computer program product according to claim 29 wherein an instantiated object model
2 adapter comprises a method for returning a reference to the distributed object together
3 with a moniker object associated with the distributed object.

1 31. Apparatus according to claim 1 wherein the first stream object comprises means
2 operable during the streaming of the distributed object out from the memory to the local
3 storage for storing the distributed object in a persistent repository that is different from
4 the local storage.

1 32. Apparatus according to claim 31 wherein the persistent repository is located remotely
2 from the local storage.

1 33 A method according to claim 11 further comprising:
2 (c) using the first stream object during the streaming of the distributed object out
3 from the memory to the local storage to store the distributed object in a persistent
4 repository that is different from the local storage.

1 34. A method according to claim 33 wherein the persistent repository is located remotely
2 from the local storage.

1 35. A computer program product according to claim 21 wherein the first stream object
2 comprises program code operable during the streaming of the distributed object out from
3 the memory to the local storage for storing the distributed object in a persistent
4 repository that is different from the local storage.

1 36. A computer program product according to claim 35 wherein the persistent repository is
2 located remotely from the local storage.